МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний технічний університет
«Харківський політехнічний інститут»

МЕТОДИЧНІ ВКАЗІВКИ
до лабораторної роботи на тему
«Організація галуження у програмах мовою С++»

з курсу «Програмування»
для студентів напряму 6.040302 – Інформатика
і курсу «Програмування та алгоритмічні мови»
для студентів напряму 6.040303 – Системний аналіз

Затверджено редакційно-видавничою
радою університету,
протокол № 3 від 28.12.09.

Харків НТУ «ХПІ» 2010

Методичні вказівки до лабораторної роботи «Організація галуження у програмах мовою С++» з курсу «Програмування» для студентів напряму 6.040302 – Інформатика і курсу «Програмування та алгоритмічні мови» для студентів напряму 6.040303 – Системний аналіз / Уклад. М. І. Безменов. – Х. : НТУ «ХПІ», 2010. – 17 с.

Укладач М. І. Безменов

Рецензент Л. М. Любчик

Кафедра системного аналізу і управління

Мета роботи

Освоєння методики організації керування процесом обчислень за допомогою операторів, що перевіряють умову.

1. ТЕОРЕТИЧНІ ОСНОВИ

1.1. Умовний оператор

Досить часто в програмі необхідно виконувати деякі дії в тому випадку, коли є істинною деяка умова. Найпростішим оператором, що використовується у цьому випадку, є оператор **if** (оператор розгалуження, умовний оператор):

```
if (вираз_умова) оператор
```

Якщо вираз_умова істинний (**true**), то виконується оператор, розташований після дужок, а потім – наступний оператор; якщо значенням виразу_умови є **false**, то оператор, що стоїть після дужок, пропускається (див. рис. 1.1, *а*). Як вираз_умова може бути записаний довільний вираз, результатом обчислення якого є одне з булевих значень **true** або **false**, а також будь-який вираз, результат обчислення якого може бути автоматично перетворений або до значення **true**, або до значення **false**. При цьому варто пам'ятати, що істинним є будь-яке значення, відмінне від нуля, а нульове значення трактується як хибне.

Наведена конструкція зручна у випадку, коли деяка дія повинна виконуватися тільки при позитивній відповіді на запитання і не виконуватися, якщо отримано негативну відповідь. Якщо ж при негативній відповіді необхідно виконати іншу дію, то оператор **if** доведеться повторити з умовою, протилежною умові, що перевіряється в першій конструкції:

```
if (вираз_умова) оператор_1
if (протилежна_умова) оператор_2
```

Більш загальною конструкцією є конструкція, ідея якої полягає в тому, щоб здійснювати перевірку не того, потрібно або не потрібно виконувати деяку дію, а того, яка із двох дій повинна бути виконана (див. рис. 1.1, *б*). При цьому обидві дії не виконуються ніколи.

Повна форма оператора **if** така:

```
if (вираз_умова) оператор_1
else оператор_2
```

Оператори, записувані після дужок, у які укладений вираз_умова, ва також після службового слова **else** (оператор, оператор_1 і оператор_2) можуть бути як

простими операторами (але не описами та визначеннями), так і складеними операторами або блоками (тобто складеними операторами, що містять описи та визначення).

Складеним оператором називається послідовність операторів, укладена у фігурні дужки:

{ оператори }

Якщо серед операторів, що утворюють складений оператор, є описи та визначення, то його називають *блоком*.

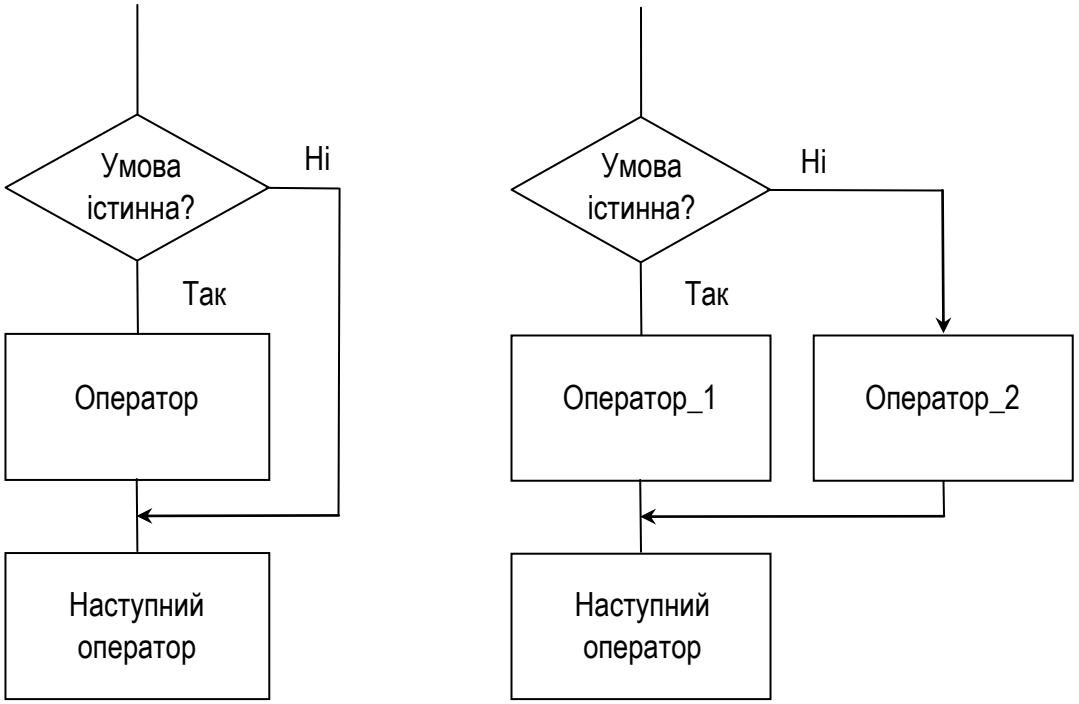

Рис. 1.1. Можливі схеми дії оператора **if**

Іноді деяку дію потрібно виконувати тільки при хибності виразу_умови. Тоді після дужок, у які укладений вираз_умова, вказують порожній оператор, позначуваний символом «точка з комою», що стоїть окремо:

```
if (вираз_умова) ;
else оператор
```

У цьому випадку більш розумно представити умову, що перевіряється, у вигляді протилежної умови:

```
if (протилежна_умова) оператор
```

Наприклад, еквівалентними за одержуваним результатом є такі дві конструкції:

```
if (x < x1) ; else z = x1;
```

та

```
if (x >= x1) z = x1;
```

Обидва оператора (оператор_1 і оператор_2), що фігурують в операторі **if**, можуть бути будь-якими операторами, у тому числі операторами, що перевіряють умову. Тому, якщо необхідно вибрати одну з декількох взаємовиключних альтернатив, використовують вкладений оператор **if**, формат якого наступний:

```
if (вираз_умова_1) оператор_1
else
  if (вираз_умова_2) оператор_2
  else оператор_3
```

або

```
if (вираз_умова_1)
  if (вираз_умова_2) оператор_1
  else оператор_2
else оператор_3
```

Відповідність між службовими словами **if** та **else** установлюється в такий спосіб: кожному **if** відповідає найближче наступне службове слово **else**, не задіяне при встановленні відповідності з іншим**if**.

1.2. Деякі особливості запису умов

Вище відзначалося, що вираз_умова може бути будь-яким виразом, тип якого може бути автоматично перетворений до типу **bool**. Найчастіше за все, поряд з арифметичними операціями, у цьому виразі використовуються операції порівняння (операції <, >, <=, >=, ==, !=), а також логічні операції ! (логічне заперечення, НЕ), && (кон'юнкція, ТА), || (диз'юнкція, АБО).

Значення виразу_умови обчислюється з урахуванням пріоритетів операцій. При цьому потрібно пам'ятати наступне:

- операція ! і операція зміни знака (унарний «мінус») мають найвищий пріоритет серед перелічених операцій (арифметичних, порівняння й логічних) і виконуються справа наліво;
- операції множення та ділення мають більш високий пріоритет у порівнянні з операціями додавання та вирахування;

- серед перелічених операцій операції `&&` та `||` мають найнижчий пріоритет;
- пріоритет арифметичних операцій вище пріоритету операцій порівняння;
- бінарні операції одного пріоритетного рівня виконуються зліва направо.

У разі необхідності зміни порядку обчислень використовуються круглі дужки.

Результат виконання логічних операцій визначається наступною таблицею:

Значення операндів		Результат		
a	b	! a	a && b	a || b
false	**false**	**true**	**false**	**false**
false	**true**	**true**	**false**	**true**
true	**false**	**false**	**false**	**true**
true	**true**	**false**	**true**	**true**

Досить часто в складних логічних виразах помилково замість операції `&&` використовується операція «кома», сенс якої полягає в тому, що кілька виразів обчислюються послідовно зліва направо, але результатом є значення самого правого виразу. Наприклад, результат обчислення виразу

```
d1 < v, v < d2
```

визначається значенням виразу `v < d2` і ніяк не залежить від значення виразу `d1 < v`. Правильною є така конструкція:

```
d1 < v && v < d2
```

В останній конструкції немає необхідності у використанні круглих дужок, оскільки операції порівняння мають більш високий пріоритет, ніж бінарні логічні операції (зокрема, `&&`).

1.3. Умовна операція

У деяких випадках оператор **`if`** може бути замінений умовною операцією `?:`, особливістю якої є те, що вона може використовуватися у виразах. Ця операція має наступний формат:

```
вираз_умова ? вираз_1 : вираз_2
```

При виконанні умовної операції в першу чергу обчислюється значення виразу_умови, що повинний бути скалярним. Якщо він істинний, то виконується обчислення значення виразу_1, а вираз_2 ігнорується. Якщо ж вираз_умова має хибне значення, то обчислюється вираз_2, а вираз_1 ігнорується. Результатом умовної операції є те зі значень вираз_1 і вираз_2, що було обчислено.

1.4. Перемикач

Перемикач (оператор багатоальтернативного вибору) використовується для організації розгалуження по багатьом альтернативам. Його формат наступний:

```
switch (вираз-перемикач)
{
    case константа_1 : оператори_1
    case константа_2 : оператори_2
      ...
    case константа_N : оператори_N
    default : оператори
}
```

Особливості складових частин оператора **switch**:

- вираз-перемикач або повинен бути цілочисловим, або допускати автоматичне перетворення свого значення до цілого типу;
- константа_1, константа_2, …, константа_N повинні бути цілочисловими, або допускати автоматичне перетворення свого значення до цілого типу;
- константа_1, константа_2, …, константа_N можуть бути іменованими константами;
- константа_1, константа_2, …, константа_N можуть бути виразами над константами;
- службове слово **case** з наступною константою, а також службове слово **default** відіграють роль міток усередині перемикача, але до цих міток не можна перейти за допомогою оператора **goto**.
- кожен з операторів, що стоять у фігурних дужках, може бути поміченим однієї або декількома конструкціями виду

case константа :

- оператори, записувані в альтернативах, можуть бути блоками.

Оператор **switch** виконується в такий спосіб.

У першу чергу здійснюється обчислення значення виразу-перемикача, що послідовно порівнюється з константами, зазначеними в альтернативах. При першому ж збігу виразу-перемикача з однієї з констант керування передається першому операторові, що стоїть за відповідною константою, після чого виконуються **всі оператори** до кінця оператора **`switch`**, якщо не передбачений примусовий вихід. Якщо значення виразу-перемикача не збігається з жодною з констант, то виконуються оператори, розташовані за міткою **`default`**, а при її відсутності в перемикачі не виконується жоден з операторів (рис. 1.2).

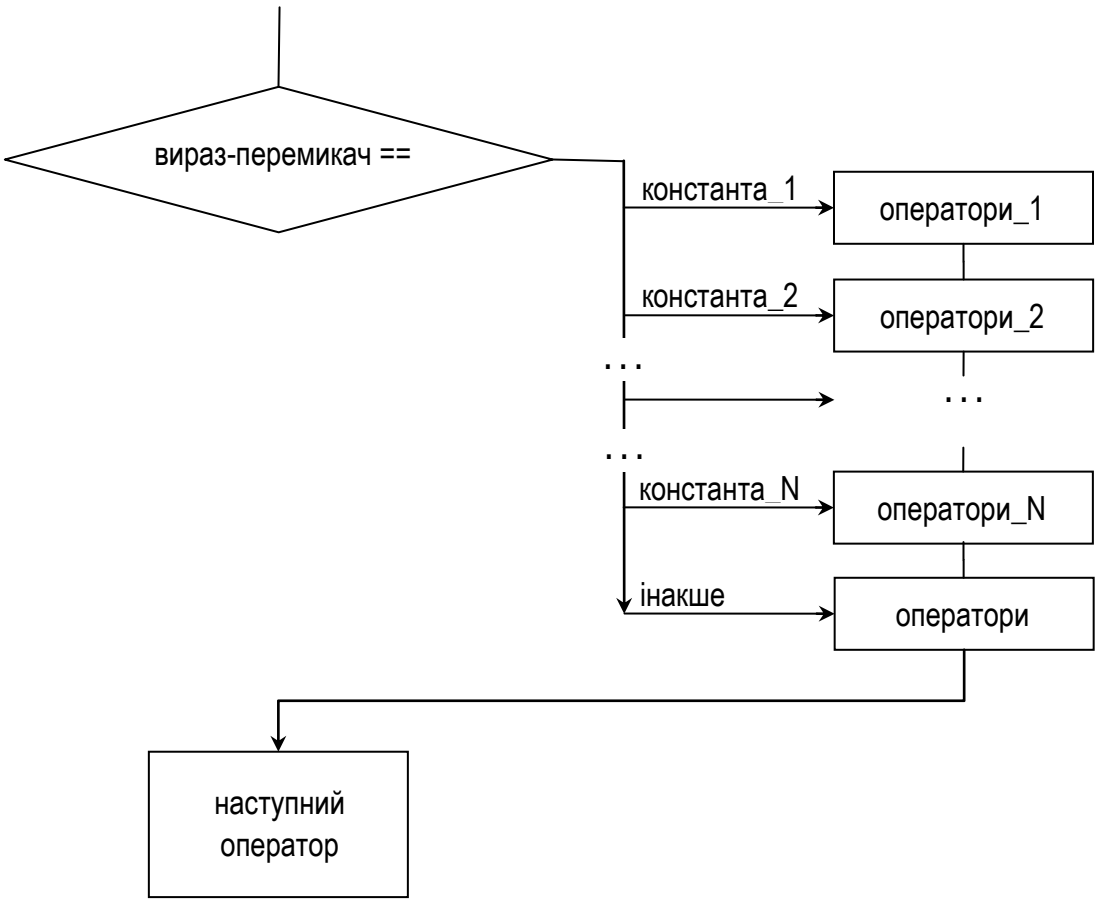

Рис. 1.2. Схема роботи оператора **`switch`**

Для забезпечення примусового виходу з перемикача використовується оператор **`break`**. Таким чином, якщо по кожній з альтернатив повинна виконатися тільки одна гілка, потрібно кожну з них (за винятком останньої) завершити оператором **`break`**.

2. ПРИКЛАДИ ПРОГРАМ

Приклад 1. Дано дійсні числа *a*, *b*, *w*. Чи правда, що на числовій осі число *w* перебуває між числами *a*, *b*?

```
#include <iostream.h>
int main()
{
   double a, b, w;

   cout << "a = ";
   cin >> a;
   cout << "b = ";
   cin >> b;
   cout << "w = ";
   cin >> w;
   cin.get();          // У потоці залишався символ кінця рядка
   if (a < w && w < b)    // Перевірка приналежності інтервалу
     cout << "Yes\n";
   else
     cout << "No\n";
   cout << "Press <Enter>";
   cin.get();
   return 0;
}
```

Приклад 2. Дано ціле число k, $|k|<1000$. Перевірити для нього справедливість правила подільності на 3. **Примітка.** Ціле число ділиться на 3 тоді і тільки тоді, коли сума його цифр ділиться на 3.

```
#include <iostream.h>
int main()
{
   int n;
   cout << "n = ";
   cin >> n;
   cin.get();          // У потоці залишався символ кінця рядка
   if (( n % 3 == 0) &&             // Якщо число ділиться на 3
       ((n % 10 +                                 // Остання цифра
         n / 10 % 10 +                    // Передостання цифра
         n / 100 % 10) % 3 != 0)         // Третя цифра з кінця
       || ( n % 3 != 0) &&       // Якщо число не ділиться на 3
       ((n % 10 + n / 10 % 10 +
         n / 100 % 10) % 3 == 0))
      cout << "The rule is not fair\n";
   else cout << "The rule is fair\n";
   cout << "Press <Enter>";
   cin.get();
   return 0;
}
```

Зауваження до програмної реалізації:

1. Оскільки правило подільності цілого числа на 3 виконується для всіх чисел програма повинна завжди видавати повідомлення `The rule is fair` (`Правило виконується`).
2. Замість цифр розглядаються остачі від ділення числа на 10 (при цьому зберігається знак).
3. За допомогою ділення числа *k* на 10 та на 100 здійснюється відсікання однієї або двох останніх його цифр відповідно.
4. Якщо число має менш трьох цифр, то результат ділення на 100 (або на 10) буде дорівнювати нулю і, як слідство, сума цифр не зміниться.

Приклад 3. Дано дійсні числа x_1, y_1, R_1, x_2, y_2, R_2, що визначають відповідно координати центрів і радіуси двох кіл на координатній площині ($R_1, R_2 \geq 0$). Чи перетинаються ці кола?

```
#include <iostream.h>
int main()
{
   double x1, y1, R1, x2, y2, R2;
   cout << "x1 = "; cin >> x1;
   cout << "y1 = "; cin >> y1;
   cout << "R1 = "; cin >> R1;
   cout << "x2 = "; cin >> x2;
   cout << "y2 = "; cin >> y2;
   cout << "R2 = "; cin >> R2;
   cin.get();          // У потоці залишався символ кінця рядка
   if ((x1 - x2) * (x1 - x2) + (y1 - y2) * (y1 - y2) <=
       (R1 + R2) * (R1 + R2)) cout << "Yes\n";
   else cout << "No\n";
   cout << "Press <Enter>";
   cin.get();
   return 0;
}
```

Приклад 4. Дано натуральне число, що не перевищує 10, яке задає числове значення оцінки. Вивести відповідну оцінку, позначену літерами: 10 – A, 9 – B, …, 6 – E, 5 і 4 – FX, 3, 2, 1 – F.

```
#include <iostream.h>
int main()
{
   int n;
   cout << "n = "; cin >> n;
   cin.get();          // У потоці залишався символ кінця рядка
   cout << n << " is equivalent ";
```

```
   switch (n)                          // Вибір одного з варіантів
   {
      case 10: cout << "A\n"; break;
      case 9:  cout << "B\n"; break;
      case 8:  cout << "C\n"; break;
      case 7:  cout << "D\n"; break;
      case 6:  cout << "E\n"; break;
      case 5: case 4:  cout << "FX\n"; break;
      case 1: case 2: case 3: cout << "F\n"; break;
      default: cout << "Error\n";
   }
   cout << "Press <Enter>";
   cin.get();
   return 0;
}
```

3. ЗАВДАННЯ НА ЛАБОРАТОРНУ РОБОТУ

За час, відведений для виконання лабораторної роботи (2 академічні години), студент повинен:

1. Розробити алгоритм розв'язання задачі, запропонованої для програмування.
2. Здійснити програмну реалізацію розробленого алгоритму.
3. Здійснити відлагодження програми, виправивши синтаксичні та логічні помилки.
4. Підібрати тестові дані для перевірки програми, включаючи виняткові випадки.
5. Оформити звіт до лабораторної роботи.
6. Відповісти на контрольні запитання.
7. Здати викладачу працездатну програму з демонстрацією її роботи на декількох варіантах вихідних даних.

4. ВАРІАНТИ ЗАДАЧ

1. Дано два дійсних числа. Вивести перше з них, якщо воно більше другого, і обидва числа, якщо це не так.
2. Дано три дійсних числа. Вибрати ті з них, які належать інтервалу (1, 3).
3. Дано дійсні значення *x*, *y*. Обчислити:

$$y = \begin{cases} x - y & \text{при } x > y, \\ y - x + 2 & \text{у супротивному випадку.} \end{cases}$$

4. Якщо сума трьох попарно різних дійсних чисел x, y, z менше одиниці, то найменше із цих трьох чисел замінити напівсумою двох інших; у супротивному випадку замінити менше з x та y напівсумою двох значень, що залишилися.
5. Дано дійсні значення x, y. Обчислити:

$$z = \begin{cases} \operatorname{arctg}\dfrac{x+y}{1-xy} & \text{при } xy < 1, \\ -\pi + \operatorname{arctg}\dfrac{x+y}{1-xy} & \text{при } xy > 1,\ x < -1, \\ -\pi + \dfrac{x+y}{1-xy} & \text{при } x < 0,\ xy > 1, \\ 1{,}57 & \text{в інших випадках.} \end{cases}$$

6. Дано дійсні числа a, b, c, d. Якщо $a \le b \le c \le d$, то кожне число замінити найбільшим з них; якщо $a > b > c > d$, числа залишити без зміни; у супротивному випадку всі числа заміняються їхніми квадратами.
7. Дано дійсні додатні числа a, b, c. З'ясувати, чи існує трикутник з довжинами сторін a, b, c. Якщо трикутник існує, то відповісти на запитання – чи є він гострокутним?
8. Дано дійсне число h. З'ясувати, чи має рівняння $ax^2 + bx + c = 0$ дійсні корені, якщо

$$a = \sqrt{\frac{|\sin 8h| + 17}{1 - \sin 4h \cos(h^2 + 18)^2}},$$

$$b = 1 - \sqrt{\frac{3}{3 + |\operatorname{tg} ah^2 - \sin ah|}},$$

$$c = ah^2 \sin bh + bh^3 \cos ah.$$

Якщо дійсні корені існують, то знайти їх. Інакше відповіддю повинне служити повідомлення, про відсутність дійсних коренів.

9. Дано дійсні числа a_1, a_2, b_2, b_1. Чи вірно, що $|a_1b_2 - a_2b_1| \ge 0.0001$? У разі позитивної відповіді знайти рішення системи рівнянь

$$a_1x + b_1y + c_1 = 0,$$

$$a_2x + b_2y + c_2 = 0.$$

10. Дано дійсні числа x_1, x_2, x_3, y_1, y_2, y_3. Чи належить початок координат трикутнику з вершинами (x_1, y_1), (x_2, y_2), (x_3, y_3)?
11. Дано додатні дійсні числа a, b, c, d. Чи можна прямокутник зі сторонами a і b повністю помістити усередині прямокутника зі сторонами c, d? Розв'язати

задачу для випадку, коли сторони одного прямокутника можуть бути тільки паралельними або перпендикулярними сторонам іншого прямокутника;.

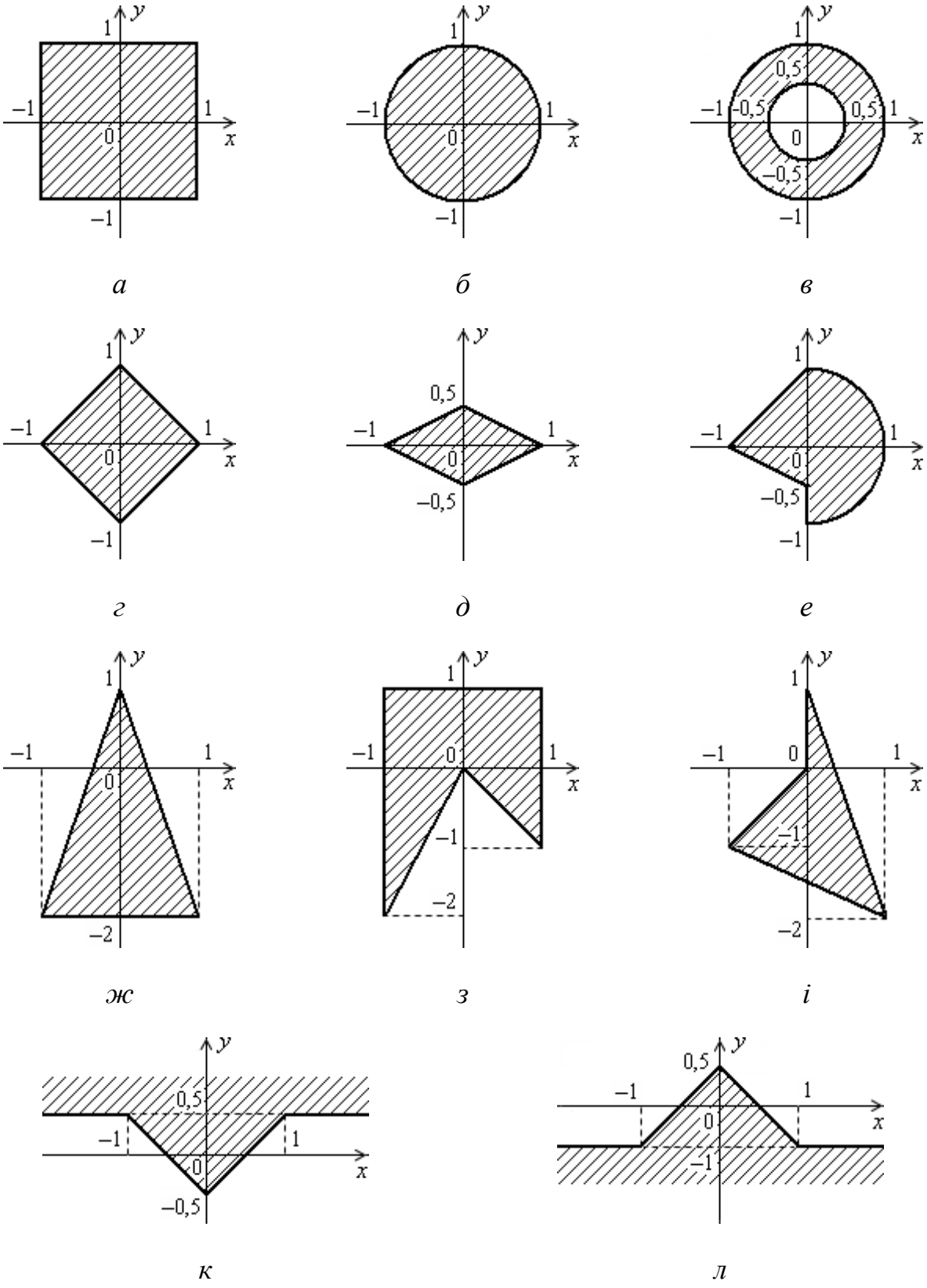

Рис. 1.3. Зв’язані області

12. Дано додатні дійсні числа *a*, *b*, *c*, *d*. Чи можна прямокутник зі сторонами *a* і *b* повністю помістити усередині прямокутника зі сторонами *c*, *d*? Розв'язати задачу для загального випадку взаємного розташування прямокутників.
13. Дано дійсні числа *x*, *y*. Визначити, чи належить точка з координатами *x*, *y* заштрихованій частині площини, включаючи її межі (рис. 1.3, *а*–1.3, *л*).
14. Дано дійсні числа *x*, *y*. Визначити, чи належить точка з координатами *x*, *y* складеній з декількох незв'язаних ділянок заштрихованій частині площини, включаючи межі (рис. 1.4, *а*–1.4, *в*).

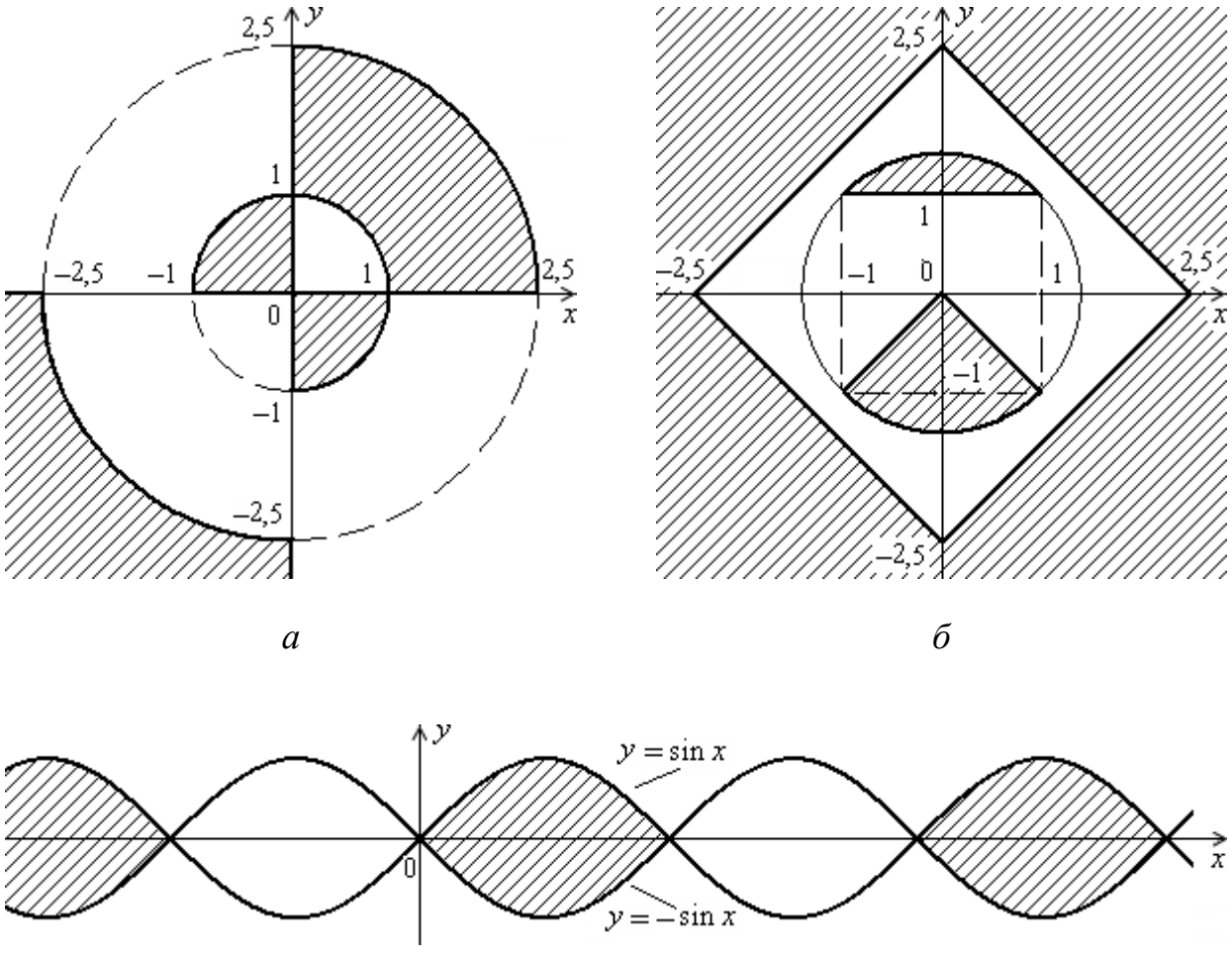

Рис. 1.3. Незв'язані області

15. Дано додатні дійсні числа *a*, *b*, *c*, що визначають розміри цегли, і додатні дійсні числа *d* і *h*, які визначають розміри прямокутного отвору. Чи пройде цегла розміру $a \times b \times c$ в прямокутний отвір розміру $d \times h$? Розв'язати задачу для довільного розташування цегли.
16. Дано додатні дійсні числа *a*, *b*, *c*, що визначають розміри цегли, і додатні дійсні числа *d* і *h*, які визначають розміри прямокутного отвору. Чи пройде

цегла розміру $a \times b \times c$ в прямокутний отвір розміру $d \times h$? Розв'язати задачу для довільного розташування цегли.

17. Дано натуральні числа k, l, p, q. Вважаючи, що в парах чисел (k, l) і (p, q) перше число є номером рядка, а друге – номером стовпця прямокутної таблиці, визначити, чи є у комірок (k, l) і (p, q) таблиці хоча б одна спільна сторона.
18. З пункту A в пункт B із швидкістю v_0 км/год виїхав автомобіль. Одночасно назустріч йому з пункту B виїхав другий автомобіль. До їх зустрічі перший автомобіль їхав з постійною швидкістю, а другий s_1 км їхав із швидкістю v_1 км/год і s_2 км – зі швидкістю v_2 км/год. Скільки годин рухалися автомобілі до їх зустрічі і яку частку загальної відстані між пунктами A і B проїхав кожний з них?
19. Пара носків коштує 1.05 грн, в'язка з 12 пар коштує 10.25 грн, а коробка з 12 в'язками коштує 114 грн. Покупцеві потрібно n пар носків. Визначити кількість коробок $n_к$, в'язок $n_в$ і пар $n_п$ носків, які повинен купити покупець, щоб потрібна кількість пар обійшлася дешевше за все.
20. Східний календар характерний 60-річним циклом. Кожний з циклів розбитий на 12-річні підцикли, які позначаються п'ятьма кольорами – зелений, червоний, жовтий, білий і чорний. Роки кожного підциклу носять назви тварин – пацюка, корови, тигра, зайця, дракона, змії, коня, вівці, мавпи, курки, собаки та свині. Початок одного із циклів (рік зеленого пацюка) був у 1924 році. Дано ціле число, що задає деякий рік. Вивести його назву за східним календарем.

5. КОНТРОЛЬНІ ЗАПИТАННЯ

1. Які оператори застосовуються для організації розгалуження?
2. У яких форматах може бути використаний оператор **`if`**?
3. Як установлюється відповідність між службовими словами **`else`** та **`if`** в операторі розгалуження?
4. Що являє собою складений оператор?
5. Що мається на увазі під терміном «блок»?
6. Чи коректним є твердження, що в програмах, написаних мовою C++, перед службовим словом **`else`** ставиться крапка з комою?
7. Як записується порожній оператор і як він використовується в операторі **`if`**?
8. Яка різниця між **`if`** `(n = 7)` та **`if`** `(n == 7)`?
9. Чому буде дорівнювати значення змінної `d` після закінчення виконання фрагмента наступного програми?

```
int a = 1, b = 1, c = 1, d = 1;
if (c = (a - b)) d = c;
```

10. Який формат має умовна операція?
11. Яке обмеження накладається на перший операнд умовної операції?
12. Коли доцільно застосовувати оператор **switch**?
13. Чи може виникнути ситуація, коли жоден з операторів в альтернативах оператора **switch** не буде виконаний?
14. Що необхідно зробити, щоб в операторі **switch** для кожної з альтернатив у разі її вибору виконувалися тільки ті оператори, які зазначені саме в ній?

СПИСОК ЛІТЕРАТУРИ

1. Страуструп, Б. Язык программирования Си++ : Второе издание / Б. Страуструп. – К. : ДиаСофт, 1993. – Ч. 1. – 264 с. ; Ч. 2. – 296 с.
2. Керниган, Б. Язык программирования Си / Б. Керниган, Д. Ритчи. – М. : Финансы и статистика, 1992. – 272 с.
3. Либерти, Джесс. Освой самостоятельно С++ за 21 день : учеб. пособ. / Джесс Либерти. – М. : Издательский дом «Вильямс», 2001. – 816 с.
4. Подбельский, В. В. Программирование на языке Си / В. В. Подбельский, С. С. Фомин. – М. : Финансы и статистика, 1999. – 600 с.
5. Подбельский, В. В. Язык Си++ / В. В. Подбельский. – М. : Финансы и статистика, 1999. – 560 с.
6. Савитч, Уолтер. Язык С++. Курс объектно-ориентированного программирования / Уолтер Савитч. – М. : Издательский дом «Вильямс», 2001. – 704 с.

Навчальне видання

Методичні вказівки
до лабораторної роботи
«Організація галуження у програмах мовою С++»
з курсу «Програмування» для студентів напряму 6.040302 – Інформатика
і курсу «Програмування та алгоритмічні мови» для студентів напряму
6.040303 – Системний аналіз

Укладач БЕЗМЕНОВ Микола Іванович

Відповідальний за випуск О. С. Куценко
Роботу до видання рекомендував О. В. Горелий

За авторською редакцією

План 2010 р., поз. 8 / 38-10

Підп. до друку 15.03.2010 р. Формат 60 × 84 $^{1}/_{16}$. Папір офісний.
Riso-друк. Гарнітура Таймс. Ум. друк. арк. 0,9. Наклад 50 прим.
Зам. № 59. Ціна договірна.

Видавничий центр НТУ «ХПІ».
Свідоцтво про державну реєстрацію ДК № 3657 від 24.12.2009 р.
61002, Харків, вул. Фрунзе, 21

Друкарня НТУ «ХПІ», 61002, Харків, вул. Фрунзе, 21